ADO.NET

كيفية التعامل مع قواعد البيانات وذلك بأستخدام مكتبيات ado.net :

- **اولا تجهيز اداة الاتصال بقاعدة البيانات وهى عباره عن جمله نصيه تكتب على حسب نوع قاعدة البيانات المستخدمه اذا كانت اكسيس او اوراكل او سيكول سيرفير .**

النوع الاول من جملة الاتصال بقاعدة البيانات وتكتب اذا كنت تعمل على سيرفير محلى والمقصود به الجهاز الذى تعمل عليه وليس سيرفير على الشبكه الدوليه

اولا استيراد فضاء الاسماء المناسب لنوع قاعدة البيانات التى تعمل عليها

```
using System.Data.SqlClient;//working with sqlserver database
using System.Data.OleDb;//working with oracle or access
using System.Data.Odbc;//working with other databse engine
```

ومن الممكن كتابى تعريفة الاتصال كبارامتر لدالة الاتصال على ان توضع فى شكل نصى

```
//تعريف كائن الاتصال بقاعدة البيانات من الفصيل المسئول عن الاتصال وهو sqlconnection
SqlConnection conn = new SqlConnection();
//اولا يتم كتابة نوع السيرفير الذى تعمل عليه اذا كان محليا اكتب اسمه او ضع نقطه
//ثانيا تكتب اسم قاعدة البيانات صحيحه
//نوع التأمين اذا كان مفتوحا والاتصال عن طريق نظام التشغيل تضع كلمة true
conn.ConnectionString = "server=.;database=Books;integrated security=true;";
//النوع الثانى مثل السابق تماما مع اختلاف ان اذا كان السيرفير له تأمين لافراد معينين
//قم بكتابة اسم المستخدم وكلمة المرور له
conn.ConnectionString = "server=.;database=Books;integrated security=false;username=ali;password=12121984";
```

اعطاء امر بفتح الاتصال بقاعدة البيانات

```
conn.Open();
```

- التعامل مع قاعدة البيانات من خلال الاوامر وذلك باستخدام كائن الفصيل SqlCommand :

ان انوع التعامل مع قاعدة البيانات كثيره جدا لكن كلها تندرج تحت صيغة اعطاء امرا ما يمرر لقاعدة البيانات برمجيا عن طريق فصيل هو المسئول عن تمرير الاوامر لقاعدة البيانات واسترجاع النتائج المطلوبه والتعامل مع هذا الفصيل لجعله ينفذ الاوامر بشكل صحيح يكون على عدة مراحل وهى كالتالى

```
//اولا يتم تعريف كائن لزر الامر
SqlCommand cm = new SqlCommand();
//يجب تعريفه بكائن الذى يقوم بالاتصال بقاعدة البيانات
cm.Connection = conn;
//النوع الاول من زر الامر ان يكون نصيا اى انك تقوم بكتابة جملة السيكول مباشرتا
cm.CommandType = CommandType.Text;
//كتابة جملة السيكول واعطائها له لكى ينفذها وهذا غير مستحب
cm.CommandText = "select * from Employees";
//النوع الثانى وهو الاجراء المخزن فيجب اعطائه اسم الاجراء المخزن داخل قاعدة البيانات
cm.CommandType = CommandType.StoredProcedure;
//اسم الاجراء المخزن داخل برنامج السيكول وهذا هو النوع الافضل
cm.CommandText = "getemployeedata";
//النوع الثالث اعطائه اسم جدول مباشى من قاعدة البيانات ليقوم ببعض العمليات عليه
cm.CommandType = CommandType.TableDirect;
//اعطائه اسم الجدول
cm.CommandText = "Employees";
//اعطاء امر لكائن الامر بتنفيذ جملة السيكول
//وهذا النوع معناه انه سيقوم بتنفيذ جمله على البيانات لا تقوم باسترجاع بيانات
//بل تقوم بحذف او ادخال او تعديل بيانات
cm.ExecuteNonQuery();
```

```
cm.ExecuteNonQuery();
//هذا النوع معناه تنفيذ القراءه بمعنى انه سيعود بكم من البيانات
//ويجب استقبالها باداه عرض للبيانات
cm.ExecuteReader();
//هذا النوع معناه ان العائد هو عباره عن رقم من انواع الارقام
int x = Convert .ToInt32 (cm.ExecuteScalar());
//هذا النوع معناه بيانات عائده لا ستقبالها فى صفحة ويب
cm.ExecuteXmlReader();
//التعامل مع الامر باعطائه بارامترات معينه والتى تتعامل مع الاجراء المخزن
//اعطائه اسم البارامتر و نوع البيانات التى يقبلها و حجم هذه البيانات
cm.Parameters.Add("@employeenames", SqlDbType.VarChar, 50);
//بعد الانتهاء من مهمة كائن الامر يجب تحرير الذاكره منه
cm.Dispose();
//ثلاث طرق لغلق عملية الاتصال تماما بقاعدة البيانات بعد الانتهاء من كل شيء
conn.Close();
//يفضل استخدامها عن السابقه
conn.Dispose();
conn = null;
//لمعرفة الحاله التى توجد عليها اداة الاتصال
//هل هى مغلقه ام مفتوحه
MessageBox.Show(conn.State.ToString());
```

مفهوم Pooling :

ومعناه تحديد عدد المتصلين بجهة اتصال واحده فى نفس الوقت وكيفية عمل ذلك داخل البرمجه

```
    //هنا يتم تعريف اكثر من جهة اتصال وسيتم اعطائها نفس تعريفة الاتصال
SqlConnection conn1 = new SqlConnection();
SqlConnection conn2 = new SqlConnection();
SqlConnection conn3 = new SqlConnection();
SqlConnection conn4 = new SqlConnection();
SqlConnection conn5 = new SqlConnection();
SqlConnection conn6 = new SqlConnection();
    //تعريف جملة اتصال واحده
    //pooling ويتم تعريف التعدد فى الاتصال او
    //ويتم اعطاء اقصى عدد ممكن لجهات الاتصال على نفس تعريفة الاتصال
string connstr = "server=.;database=Books;integrated security=true;pooling=true;max pool site=5";
    //اعطاء نفس جملة الاتصال لجميع جهات الاتصال المختلفه
conn1.ConnectionString = connstr;
conn2.ConnectionString = connstr;
conn3.ConnectionString = connstr;
conn4.ConnectionString = connstr;
conn5.ConnectionString = connstr;
conn6.ConnectionString = connstr;

   //بعد ذلك قم بفتح جميع جهات الاتصال مره واحده كل واحده فى زر امر مختلف
    //وتقوم باختبار حالتها
    //لاحظ انك لا تستطيع فتح اتصال اكبر من العدد الذى قمت بتحديده فى تعريفة الاتصال
conn1.Open();
```

```
         //بعد ذلك قم بفتح جميع جهات الاتصال مره واحده كل واحده فى زر امر مختلف
         //وتقوم باختبار حالتها
          //لاحظ انك لا تستطيع فتح اتصال اكبر من العدد الذى قمت بتحديده فى تعريفة الاتصال
      conn1.Open();
      MessageBox.Show(conn1.State.ToString());
      conn2.Open();
      MessageBox.Show(conn2.State.ToString());
      conn3.Open();
      MessageBox.Show(conn3.State.ToString());
      conn4.Open();
      MessageBox.Show(conn4.State.ToString());
      conn5.Open();
      MessageBox.Show(conn5.State.ToString());
      conn6.Open();
      MessageBox.Show(conn6.State.ToString());
        }


    }
}
```

- قراءة البيانات والتعامل مع Data Reader داخل برمجة قواعد البيانات :

ويتعامل كائن Data Reader مع قاعدة البيانات بهدف قراءة المحتويات او استرجاع البيانات المطلوبه ولا تقوم بانشاء كائن جديد منه عند التعامل هو فقط يحتاج الى تعريف متغير وينفذ بواسطة كائن sqlcommand ولنرى المثال التالى :

```
private void button1_Click(object sender, EventArgs e)
   اولا يتم تعريف جملة الاتصال وكتابة اسم السيرفير المحلى//
{SqlConnection cn = new SqlConnection("server=SOFYANY-PC\\SQLEXPRESS;database=Books;integrated security=true");
   تعريف كائن تنفيذ الاوامر//
SqlCommand cm = new SqlCommand();
   تعريف قارىء البيانات ولا يحتاج الا لتعريف متغير فقط منه//
SqlDataReader dr;
   تعريف كائن جدول البيانات والذى يقوم بتخزين البيانات//
   التى تأتى من جملة الاستعلام داخله//
DataTable dt = new DataTable();
try
{ //فتح الاتصال
   cn.Open();
   cm.Connection = cn;
   cm.CommandText = "select * from Authors";//جملة الاستعلام
   dr = cm.ExecuteReader();//تنفيذ الامر عن طريق كائن الاوامر ثم اعطاء النتيجه لمتغير قارى البيانات
   dt.Load(dr);//ثم تحميل البيانات من القارىء الى كائن الجدول
   dataGridView1.DataSource = dt;//ثم اعطاء ما تم تخزينه الى مصدر بيانات شبكة البيانات
}catch(Exception ex){MessageBox.Show(ex.Message);}
finally{cn.Close();cn.Dispose();}}
```

النقطه الثانيه عندما نريد قراءة بيانات عمود واحد او حقل واحد من البيانات فاننا نستخدم اداتى

List Box – Combo Box

ولو استخدمنا المثال السابق فى قراءة حقل واحد من جدول Authors وبنفس الكود مع اختلاف وسيلة عرض البيانات فانه غالبا ما يتم استخدام الحلقات فى ذلك

```
    //تعريف قارىء البيانات ولا يحتاج الا لتعريف متغير فقط منه
SqlDataReader dr;
try
{ //فتح الاتصال
    cn.Open();
    cm.Connection = cn;
    cm.CommandText = "select firstname from Authors";//جملة الاستعلام
    dr = cm.ExecuteReader();//تنفيذ الامر عن طريق كائن الاوامر ثم اعطاء النتيجه لمتغير قارى البيانات
    //ثم الدخول فى حلقه لقراءة الحقل سجل تلو الاخر حتى النهايه
    while (dr.Read())
    {
        //اضافة ما يتم قرائته من سجلات الى عناصر القائمه
        //ويتم وضع الفهرس من الصفر تصاعديا
        //او من الممكن ان تكتب اسم الحقل المراد قرائته
        listBox1.Items.Add(dr["firstname"].ToString());
    }
}catch(Exception ex){MessageBox.Show(ex.Message);}
finally{cn.Close();cn.Dispose();}}
```

النقطه الثالثه وهى عملية ادخال بيانات الى الجدول والتى تكون مختلفه فقط فى جملة SQL بالاضافه اعطاء الامر بعدم استرجاع البيانات وانما ادخالها فقط وهو ExcuteNonQuery ولنرى مثال

```
try
{ //فتح الاتصال
    cn.Open();
    cm.Connection = cn;
    //التأكد اولا من ان صناديق الادخال ليست فارغه
    if (FIRSTNAMETXT.Text != "" && LASYNAMETXT.Text != "")
        //كتابة جملة الادخال وتكون على هذه الطريقه
    { cm.CommandText = "INSERT INTO Authors (firstname,lastname)VALUES('" + FIRSTNAMETXT.Text + "','"
            + LASYNAMETXT.Text + "')";
        //تنفيذ الامر بهدف ادخال بيانات فقط دون استرجاع شىء
        cm.ExecuteNonQuery(); }
    //تنبيه للمستخدم بنجاح عملية الادخال
    statuslb.Text = "Insert command Successful" + cm.CommandText.ToString();
}
    //اظهار الاخطاء ان وجدت
catch (sqlException ex) { MessageBox.Show(ex.Message); Console.WriteLine(ex.InnerException.ToString()); }
finally { cn.Close(); cn.Dispose(); cm.Dispose(); }
}
```

استخدام امر التحديث او Update والامر يكون مختلف فقط عن السابق فى جملة SQL وبنفس تنسيق الاوامر السابق لكن يجب التحديث على اساس قيمه لا تتكرر فى جميع سجلات الجدول فمثلا نختار التحديث على اساس حقل المفتاح الاساسى وهذا هو الاضمن ولنرى المثال

```
try
{ //فتح الاتصال
    cn.Open();
    cm.Connection = cn;
    //التأكد اولا من ان صندوق الذى يكون على اساسه التحديث ليس فارغا
    if (authoridtxt .Text != "" && FIRSTNAMETXT.Text != "" && LASYNAMETXT .Text != "")
        //كتابة جملة الادخال وتكون على هذه الطريقه
    { cm.CommandText = "UPDATE Authors SET " +
          "firstname = '" + FIRSTNAMETXT.Text +
          "',lastname = '" + LASYNAMETXT.Text +
          "' WHERE authorid = '" + authoridtxt.Text + "'";
        //تنفيذ الامر بهدف ادخال بيانات فقط دون استرجاع شيء
        cm.ExecuteNonQuery(); }
    //افراغ الحقول بعد التاكد من نجاح عملية الادخال
    authoridtxt.Text = "";
    FIRSTNAMETXT.Text = "";
    LASYNAMETXT.Text = "";
    //تنبيه للمستخدم بنجاح عملية الادخال
    statuslb.Text = "Update command Successful" + cm.CommandText.ToString(); }
    //اظهار الاخطاء ان وجدت
catch (sqlException ex) { MessageBox.Show(ex.Message); Console.WriteLine(ex.InnerException.ToString()); }
finally { cn.Close(); cn.Dispose(); cm.Dispose(); }
}
```

استخدام امر الحذف Delete داخل قاعدة البيانات والاختلاف ايضا عن امر التحديث هو فى جملة SQL ويجب ان يكون الحذف بمعلومية حقل لا يتكرر على مستوى الجدول وليكن المفتاح الاساسى المثال :

```
    SqlCommand cm = new SqlCommand();
    try
    { //فتح الاتصال
        cn.Open();
        cm.Connection = cn;
        //التأكد اولا من ان صندوق الذى يكون على اساسه الحذف ليس فارغا
        if (authoridtxt .Text != "")
            //كتابة جملة الحذف وتكون على هذه الطريقه
        { cm.CommandText = "DELETE FROM Authors WHERE authorid = '" + authoridtxt .Text + "'";
            //تنفيذ الامر بهدف حذف بيانات فقط دون استرجاع شيء
            cm.ExecuteNonQuery(); }
        //تنبيه للمستخدم بنجاح عملية الحذف
        statuslb.Text = "Delete command Successful" + cm.CommandText.ToString(); }
        //اظهار الاخطاء ان وجدت
    catch (sqlException ex) { MessageBox.Show(ex.Message); Console.WriteLine(ex.InnerException.ToString()); }
    finally { cn.Close(); cn.Dispose(); cm.Dispose(); }
    }
  }
}
```

نصل الان الى موضوع هام وهو Transaction ويستخدم فى حالة تنفيذ عدد من العمليات المتتابعه بفرض انه يجب تنفيذها جميعا واذا فشلت احداهما لا تنفذ التى تليها وهكذا وعادة ما تكون تلك العمليات مرتبطه مع بعضها بحيث تقوم بتنفيذ وظيفه واحده لكن على مراحل كالاستعلام عن سجل داخل جدول باحد حقوله استعدادا لحذفه من الجدول او تحديثه مثلا ولنرى مثال :

```
public partial class Form1 : Form
{ SqlConnection cn = new SqlConnection("server=SOFYANY-PC\\SQLEXPRESS;database=Books;integrated security=true"
  SqlCommand cm1 = new SqlCommand();
  SqlCommand cm2 = new SqlCommand();
  SqlDataReader dr;
    public Form1()
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)

    {try{ cn.Open();
        //التعريف بعملية transaction
          SqlTransaction sqltr = null;cm1.Connection = cn;cm2.Connection = cn;
        //اعطاء الامر ببداية العمليه
            sqltr = cn.BeginTransaction();
        //تعريف كائن الامر على متغير العمليه
            cm1.Transaction = sqltr;cm2.Transaction = sqltr;
            cm1.CommandText = "SELECT * FROM Authors WHERE authorid = '" + authoridtxt.Text + "'";
            dr = cm1.ExecuteReader();



    dr = cm1.ExecuteReader();
//اختبار هل يوجد بيانات يقوم القارىء بقرائتها
    if (dr.Read()!= false)
    { //غلق القارىء
     dr.Close();
     //ثم تنفيذ العمليه الثانيه
    cm2.CommandText = "DELETE FROM Authors WHERE authorid = '" + authoridtxt.Text + "'";
    cm2.ExecuteNonQuery();
    statuslb.Text = "Full Operation Succesful" + cm1.CommandText.ToString() + cm2.CommandText.ToString();
    authoridtxt.Text = ""; }
    else
    { //اما اذا لم يوجد سجل برقم المؤلف فيتم الغاء العمليه الثانيه
        sqltr.Rollback();} }
catch (SqlException ex)
{Console.WriteLine(ex.InnerException.ToString());
 MessageBox.Show(ex.Message); }
finally
{ dr.Dispose(); cm1.Dispose(); cm2.Dispose();}}}}
```

مفهوم Disconnected Mode وما الفرق بينه وبين Connected Mode :

كل ماسبق برمجته كان فى اطار الاتصال المباشر مع قاعدة البيانات اى ان جميع العمليات السابقه تنفذ مع استمرار الاتصال بقاعدة البيانات ويسمى Connected Mode لكن الوضع الثانى هو كيفية استدعاء البيانات فى وضع عدم الاتصال او Disconnected Mode

استخدام الوضع الاول فى حالة العمل على جهاز محلى بقاعدة بيانات واحده تتاح لمستخدم واحد فيفضل هنا الاتصال المستمر بقاعدة البيانات

لكن الوضع الثانى يفضل فى حالة وجود شبكة بيانات تتاح لاكثر من مستخدم لادخال البيانات والخلاصه انك تقوم بكتابة او تعديل او حذف ما تريده بيحث يسجل كل هذا فى جدول وهمى ثم بعد ان تتأكد من كل تلك التعديلات تقوم بفتح قاعدة البيانات لتطبق فيها كل ما تم تغييره

فالنوع الثانى يتعامل مع عنصريين اساسين هما Data Adapter و Dataset والاخير هو عباره عن جدول وهمى الذى يتم التعامل معه بعيدا عن قاعدة البيانات او يمثل صوره كامله منها

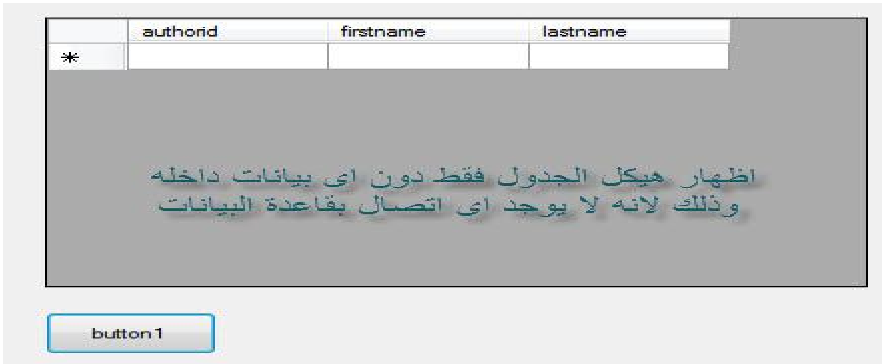
وظيفة Data Adapter ان يقوم بدور الوسيط بين قاعدة البيانات وDataset فى تبادل المعلومات

ولنرى المثال التالى :

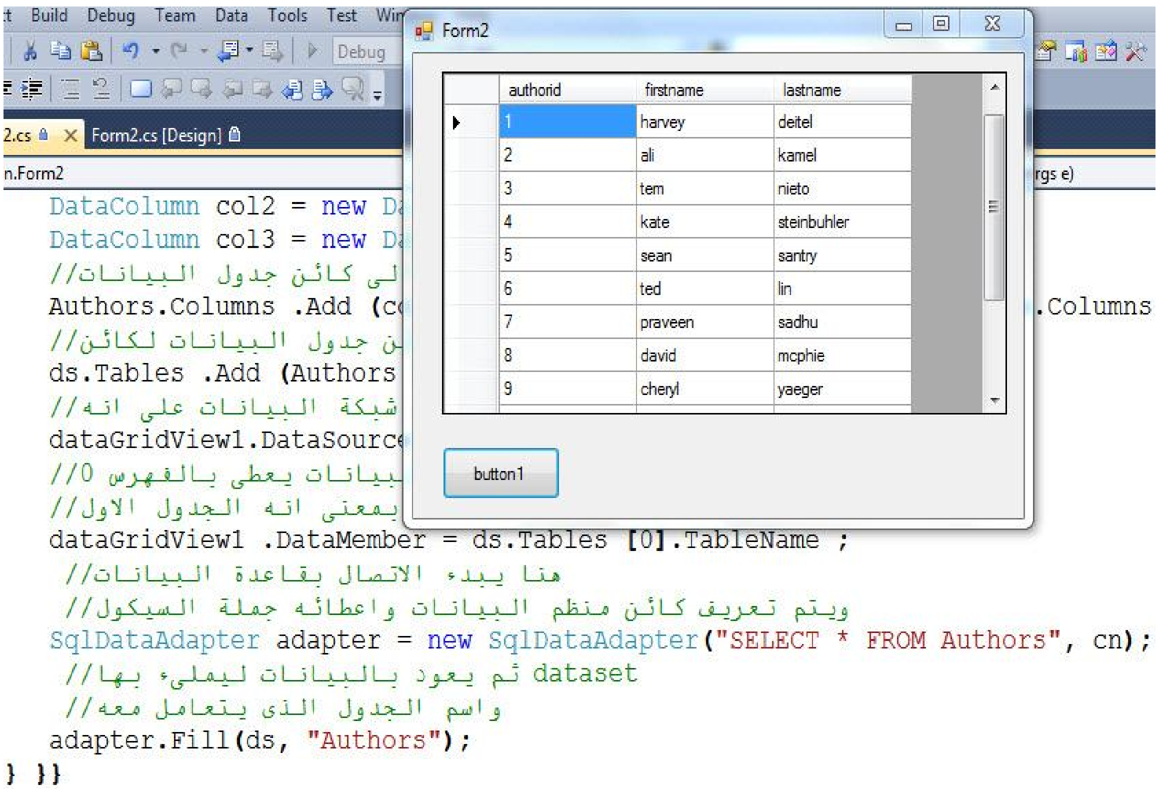
```
private void button1_Click(object sender, EventArgs e)
{ //تعريف متغير جديد من dataset
    DataSet ds = new DataSet();
    //تعريف متغير من كائن جدول البيانات واعطاؤه اسم الجدول المطلوب
    DataTable Authors = new DataTable("Authors");
    //تعريف متغير من حقل البيانات واعطاؤه اسم الحقل ثم نوع بياناته
    DataColumn col1 = new DataColumn("authorid",typeof(int));
    //هذا بالنسبه لحقل المفتاح الاساسى انه يزيد بشكل تلقائى
    col1.AutoIncrement = true;
    //مقدار الزياده 1 والبدايه من 1
    col1.AutoIncrementSeed = 1; col1.AutoIncrementStep = 1;
    //تعريف باقى الحقول
    DataColumn col2 = new DataColumn ("firstname",typeof(string));
    DataColumn col3 = new DataColumn ("lastname",typeof(string));
    //اضافة ما تم تعريفه من حقول الى كائن جدول البيانات
    Authors.Columns .Add (col1); Authors.Columns .Add (col2); Authors.Columns .Add (col3);
    //ثم اعطاء كائن جدول البيانات لكائن dataset
    ds.Tables .Add (Authors );
    //التعيرف بمصدر بيانات شبكة البيانات على انه dataset
    dataGridView1.DataSource = ds ;
    //التعريف بان الجدول الذى تحمله شبكة البيانات يعطى بالفهرس 0
    //بمعنى انه الجدول الاول
    dataGridView1 .DataMember = ds.Tables [0].TableName ; } }}
```

(local variable) DataTable Authors

وعندما ترى تنفيذ البرنامج فيصبح بهذا الشكل :



هذه هى النتيجه بعد الاتصال بقاعدة البيانات



ان استخدام اسلوب Disconnected يمثل عبىء كبير فى كتابة الكود لكن يجب العمل به اذا كان قاعدة البيانات على شبكه تتعامل مع اكثر من مستخدم فى نفس الوقت

استخدام LINQ فى الاستعلامات داخل قاعدة البيانات :

لغة مبتكره فى الدوت نت الاصداره 3 وهى البديل عن لغة الاستعلامات SQL باختلافات بسيطه فى طريقة كتابة الكود لكن تقدم تسهيلات كبيره من ناحية الاتصال والاستعلام عن البيانات

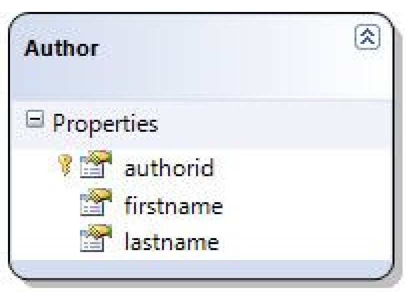
خطوات الاستخدام لهذه التقنيه :

1 – ابدء مشروعك الجديد وقم بوضع ما تريد من ادوات التعامل مع قواعد البيانات وليكن DataGride

2 – تذهب الى نافذة المشروع وتقوم باضافة عنصر جديد وهو



2 – تقوم بفتح نافذة Server وتختار منها الجدول المطلوب وتضعه على صفحة linq



فيقوم ملف Design لتلك الاداه بتكويد كل ما يلزم لاستخدام الاداه داخل قاعدة البيانات وداخل الجدول الذى تم تحديده

- استخدام Query Expression بديلا عن اوامر SQL

وتستخدم ما يسمى بالمتغيرات مجهولة النوع اى لاتعرف ماهو نوع البيانات الذى يجب ان يستخدم

وتعرف بأسم Anonymous type ولنرى مثال :

Select by Linq – 1

```
private void button1_Click(object sender, EventArgs e)
{
    //اولا يجب اخذ كائن من الملف الخاص linq
    //والذى تم انشاؤه ووضع الجدول فيه
    //ولاحظ انه دائما يضاف الى اسمه context
    DataClasses1DataContext db = new DataClasses1DataContext();
    //الخطوه الثانيه وهى كتابة عبارة الاستعلام
    //والتى تكون عباره عن query expression
    //والتى تبدء بتعريف متغير مجهول لا يعرف نوع بيانته
    //وجملة الاستعلام هى التى ستحدد بياناته
    //الجمله الاتيه معناها استدعى كافة الحقول
    // لاحظ هنا مدى السهوله حيث انك تختار الجداول كما انك تستطيع اختيار الحقول ايضا
    var T = from P in db.Authors
            select P;
    //وبكل بساطه تم اعطاء قيمة المتغير كمصدر بيانات لسبكة البيانات
    dataGridView1.DataSource = T;
}
```

ماذا لو اردنا اظهار حقول بعينها دون الاخرى ماذا يكون شكل جملة الاستعلام

```
var T = from P in db.Authors
        //بهذه الطريقه يتم انتقاء الحقول التى تريدها
        select new { P.firstname, P.lastname };
//وبكل بساطه تم اعطاء قيمة المتغير كمصدر بيانات لسبكة البيانات
dataGridView1.DataSource = T;
```

اما اذا اردت وضع شرط للاستعلام فيكون كالتالى :

```
var T = from P in db.Authors
        //وضع شرط للاستعلام
        where P.authorid == 3
        //بهذه الطريقه يتم انتقاء الحقول التى تريدها
        select new { P.firstname, P.lastname };
//وبكل بساطه تم اعطاء قيمة المتغير كمصدر بيانات لسبكة البيانات
dataGridView1.DataSource = T;
```

استخدام جمل الترتيب :

```
var T = from P in db.Authors
        //وضع شرط للاستعلام
        orderby P.authorid ascending
        //بهذه الطريقه يتم انتقاء الحقول التى تريدها
        select new { P.firstname, P.lastname };
//وبكل بساطه تم اعطاء قيمة المتغير كمصدر بيانات لسبكة البيانات
dataGridView1.DataSource = T;
```

لاحظ استخدام الترتيب

لاحظ ان جميع الكلمات المستخدمه داخل السيكول معرفه داخل linq

Insert By Linq – 2

Object Initialization Expression

لابد عند التعامل مع قواعد البيانات ان تقوم بعمل Class لكل جدول من قاعدة البيانات

من الملاحظ انه عند استخدام LinqClass فقد بنء بنفسه Class تابع لقاعدة البيانات ووضع فيه كافة الحقول




```
[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.Authors")]
public partial class Author : INotifyPropertyChanging, INotifyPropertyChanged
{

    private static PropertyChangingEventArgs emptyChangingEventArgs = new Prop

    private int _authorid;

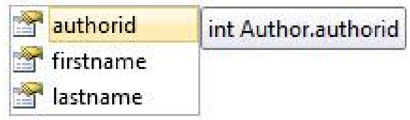
    private string _firstname;

    private string _lastname;
```

فمن السهل جدا استدعاء تلك الخصائص والتى تمثل كافة حقول الجدول

```
public class Authors
{
        int authorid;
        public int Authorid
        {
                get { return authorid; }
                set { authorid = value; }
        }
        string firstname;
        public string Firstname
        {
                get { return firstname; }
                set { firstname = value; }
        }
        string lastname;
        public string Lastname
        {
                get { return lastname; }
                set { lastname = value; }
        }
```

```
private void button2_Click(object sender, EventArgs e)
{
    Author au = new Author {a };
}
```




```
private void button2_Click(object sender, EventArgs e)
{
    //هذه الجمله هى ما يعرف بى//
    //object inialization
    //دالة فى للحقول القيم باعطاء تقوم ان وهى constructor
    Author au = new Author {firstname = "sami",lastname = "ahmed"};
    //ملف من كائن بانشاء تقوم ثم linq
    DataClasses1DataContext db = new DataClasses1DataContext();
    //تقوم باستدعاء الجدول ومنه دالة ادخال البيانات والتى//
    //تكون بهذا الشكل//
    db.Authors.InsertOnSubmit(au);
    //التأكيد على التغيير بادخال بيانات جديده//
    db.SubmitChanges();
}
```

- طريقة تحديث البيانات :

وتعريف lambda Expression

تستخدم فى عملية تسهيل الكتابه بأنشاء تعبير رمزى يعبر عن الجدول الذى تعمل عليه ولنرى مثال

```
//ملف من كائن بانشاء نقوم dbml
DataClasses1DataContext db = new DataClasses1DataContext();
//تعبير انشاء طريقة lambda
//رمز ان p
//يمكن ان يكون اى رمز تختاره لانه فى النهايه عائد على الجدول//
Author au = db.Authors.First(p => p.firstname.StartsWith("ali"));
//ادخال البيانات الجديده//
au.firstname = "sami";
au.lastname = "khaled";
//تثبيت التغيير//
db.SubmitChanges();
```

الحذف باستخدام نفس التعبير السابق

وسيكون نفس الكود السابق لكن مع تغير بسيط فى الامر وهو الحذف بدل تاكيد التعديل ثم تاكيد التعديل

```
private void button2_Click(object sender, EventArgs e)
{
    //نقوم بانشاء كائن من ملف dbml
    DataClasses1DataContext db = new DataClasses1DataContext();
    //طريقة انشاء تعبير lambda
    //ان رمز p
    //يمكن ان يكون اى رمز تختاره لانه فى النهايه عائد على الجدول
    Author au = db.Authors.First(p => p.firstname.StartsWith("tem"));
    //حذف محتوى الاستعلام الذى يعود به المتغير au
    //اى انه يقوم بحذف الصف الذى الاسم الاول له تيم
    db.Authors.DeleteOnSubmit(au);
    //تثبيت التغيير
    db.SubmitChanges();
```

****

****

- ننتقل الان الى التعريف بأهم اوامر SQL وفيما تستخدم :

SELECT * FROM *tableName*　　　جملة الاستعلام العامه



```
SELECT fieldName1, fieldName2, … FROM tableName WHERE criteria
```
فورم جملة الاستعلام

```
SELECT title, editionNumber, copyright
FROM Titles
WHERE copyright > 1999
```
مثال على استخدام الشرط

معاملات الشروط التى تستخدم

The **WHERE** clause condition can contain operators <, >, <=, >=, =, <> and **LIKE**. Operator **LIKE** is used for *pattern matching* with wildcard characters *asterisk* (*) and *question*

```
SELECT authorID, firstName, lastName
FROM Authors
WHERE lastName LIKE 'D*'
```
احضار الاسماء التى تبدء بحرف D

```
SELECT authorID, firstName, lastName
FROM Authors
WHERE lastName LIKE '?i*'
```
هنا يبحث عن الاسماء التى الحرف i فيها هو الحرف الثانى

```
SELECT fieldName1, fieldName2, … FROM tableName ORDER BY field ASC
SELECT fieldName1, fieldName2, … FROM tableName ORDER BY field DESC
```

```
SELECT authorID, firstName, lastName
FROM Authors
ORDER BY lastName DESC
```
ترتيب تنازليا

```
SELECT authorID, firstName, lastName
FROM Authors
ORDER BY lastName ASC
```
ترتيب تصاعديا

وممكن الترتيب بواسطة اكثر من حقل والترتيب الاساسى هو التصاعدى

من الممكن اجتماع الشرط والترتيب فى جمله واحده

استرجاع جميع العنواين التى تنتهى بالعباره المكتوبه

```
SELECT isbn, title, editionNumber, copyright, price
FROM Titles
WHERE title
LIKE '*How to Program' ORDER BY title ASC
```

استرجاع المعلومات بعلاقة الترابط الداخلى بين المفتاح الاجنبى والاساسى

```
SELECT fieldName1, fieldName2, …
FROM table1
INNER JOIN table2
    ON table1.fieldName = table2.fieldName
```

```
SELECT firstName, lastName, isbn
FROM Authors
INNER JOIN AuthorISBN
    ON Authors.authorID = AuthorISBN.authorID
ORDER BY lastName, firstName
```

الربط بين اكثر من جدولين

لاحظ كتابة جدول كل حقل من الحقول

ربط الجزء الاول والذى يسترجع منه يتم ربطه بالمستنتج من الربط الذى يليه

```
SELECT Titles.title, Titles.isbn, Authors.firstName,
       Authors.lastName, Titles.copyright,
        Publishers.publisherName
FROM
   ( Publishers INNER JOIN Titles
      ON Publishers.publisherID = Titles.publisherID )
      INNER JOIN
```

Fig. 19.22 **TitleAuthor** query of **Books** database. (Part 1 of 2.)

Chapter 19 Database, SQL and ADO .NE

الجزء الثانى من الربط

ربط الاستنتاجين معا بهذا الشرط

```
      ( Authors INNER JOIN AuthorISBN
      ON Authors.authorID = AuthorISBN.authorID )
   ON Titles.isbn = AuthorISBN.isbn
ORDER BY Titles.title
```

Insert Statement

9.4.6 INSERT Statement

'he ***INSERT*** statement inserts a new record in a table. The simplest form for this stat

```
INSERT INTO tableName ( fieldName1, fieldName2, …, fieldNameN )
   VALUES ( value1, value2, …, valueN )
```

```
INSERT INTO Authors ( firstName, lastName )
   VALUES ( 'Sue', 'Smith' )
```

Update Statement

.7 UPDATE Statement

PDATE statement modifies data in a table. The simplest form for an **UPDATE** state-
is:

```
UPDATE tableName
   SET fieldName1 = value1, fieldName2 = value2, …, fieldNameN = valueN
   WHERE criteria
```

```
UPDATE Authors
   SET lastName = 'Jones'
   WHERE lastName = 'Smith' AND firstName = 'Sue'
```

Delete statement

19.4.8 DELETE Statement

An SQL ***DELETE*** statement removes data from a
statement is:

```
DELETE FROM tableName WHERE criteria
```

```
DELETE FROM Authors
   WHERE lastName = 'Jones' AND firstName = 'Sue'
```

Sofyany

هذا الكتاب ضمن سلسله من الكتب التى يتم تحمليها الى موقع الحاسب العربى

ايميل السابق

Memorycode_84@yahoo.com

الحالى

codacso@yahoo.com